Stefan **Bold**, Benedikt **Löwe**, Thoralf **Räsch**, Johan **van Benthem** (*eds.*)
Infinite Games
Papers of the conference "*Foundations of the Formal Sciences V*"
held in Bonn, November 26-29, 2004

An ω-power of a context-free language which is Borel above Δ^0_ω

Jacques Duparc and Olivier Finkel
Université de Lausanne,
Information Systems Institute, and
Western Swiss Center for Logic, History and Philosophy of Sciences
Bâtiment Provence
CH-1015 Lausanne

and

Equipe Modèles de Calcul et Complexité
Laboratoire de l'Informatique du Parallélisme **
CNRS et Ecole Normale Supérieure de Lyon
46, Allée d'Italie 69364 Lyon Cedex 07, France.
E-mail: jacques.duparc@unil.ch and Olivier.Finkel@ens-lyon.fr

Abstract. We use erasers-like basic operations on words to construct a set that is both Borel and above Δ^0_ω, built as a set V^ω where V is a language of finite words accepted by a pushdown automaton. In particular, this gives a first example of an ω-power of a context free language which is a Borel set of infinite rank.

1 Preliminaries

Given a set A (called the alphabet) we write A^*, and A^ω, for the sets of finite, and infinite words over A. We denote the empty word by ϵ. In

** UMR 5668 - CNRS - ENS Lyon - UCB Lyon - INRIA
LIP Research Report RR 2007-17
Received: ...;
In revised version: ...;
Accepted by the editors: ....
2000 *Mathematics Subject Classification.* **PRIMARY** SECONDARY.

order to facilitate the reading, we use u, v, w for finite words, and x, y, z for infinite words. Given two words u and v (respectively, u and y), we write uv (respectively,uy) for the concatenation of u and v (respectively, of u and y). Let $U \subseteq A^*$ and $Y \subseteq A^* \cup A^\omega$, we set: $UY = \{uv, uy : u \in X \wedge v, y \in Y\}$.

We recall that, given a language $V \subseteq A^*$, the *ω-power* of this language is

$$V^\omega = \{x = u_1 u_2 \dots u_n \dots \in A^\omega : \forall n < \omega \; u_n \in V \setminus \{\varepsilon\}\}$$

A^ω is equipped with the usual topology. i.e. the product of the discrete topology on the alphabet A. So that every open set is of the form WA^ω for any $W \subseteq A^*$. Or, to say it differently, every closed set is defined as the set of all infinite branches of a tree over A. We work within the Borel hierarchy of sets which is the strictly increasing (for inclusion) sequence of classes of sets $(\Sigma^0_\xi)_{\xi<\omega_1}$ - together with the dual classes $(\Pi^0_\xi)_{\xi<\omega_1}$ and the ambiguous ones $(\Delta^0_\xi)_{\xi<\omega_1}$ - which reports how many operations of countable unions and intersections are necessary to produce a Borel set on the basis of the open ones.

A reduction relation between sets $X,\ Y$ is a partial ordering $X \leq Y$ which expresses that the problem of knowing whether any element x belongs to X is at most as complicated as deciding whether $f(x)$ belongs to Y, for some given *simple* function f. A very natural reduction relation between sets of infinite words (closely related to reals), has been thoroughly studied by Wadge in the seventies. From the topological point of view, *simple* means continuous, therefore the Wadge ordering compares sets of infinite sequences with respect to their fine topological complexity. Associated with determinacy, this partial ordering becomes a prewellordering with anti-chains of length at most two. The so called Wadge Hierarchy it induces incredibly refines the old Borel Hierarchy. Determinacy makes it way through a representation of continuous functions in terms of strategies for player II in a suitable two-player game: the Wadge game $W(X, Y)$. In this game, players I and II, take turn playing letters of the alphabet corresponding to X for I, and letters of the alphabet corresponding to Y for II. In order to get the right correspondence between a strategy for player II and a continuous function, player II is allowed to skip, whereas I is not. However, II must play infinitely many letters.

As usual, reduction relations induce the notion of a complete set: a set that both belongs to some class, whose members it also reduces. In the

context of Wadge reducibility, a set is complete if it belongs to some class closed by inverse image of continuous functions, and reduces everyone of its members. A class which admits a complete set is called a Wadge Class. As a matter of fact, all Σ^0_ξ, and Π^0_ξ, are Wadge classes, whereas Δ^0_ξ ($\xi > 1$) are not.

For instance, the set of all infinite sequences that contains a 1 is Σ^0_1-complete, the one that contains infinitely many 1s is Π^0_2-complete. As a matter of fact, reaching complete sets for upper levels of the Borel hierarchy, requires other means which we introduce in next sections.

2 Erasers

For climbing up along the finite levels of the Borel hierarchy, we use erasers-like moves, see [Dup01]. For simplicity, imagine a player (either I or II) playing a Wadge game, in charge of a set $X \subseteq A^\omega$, with the extra possibility to delete any terminal part of her last moves.

We recall the definition of the operation $X \mapsto X^{\approx}$ over sets of infinite words. It was first introduced in [Fin01] by the second author, and is a simple variant of the first author's operation of exponentiation $X \mapsto X^{\sim}$ which first appeared in [Dup01].

We denote $|v|$ the length of any finite word v. If $|v| = 0$, v is the empty word. If $v = v_1 v_2 \dots v_k$ where $k \geq 1$ and each v_i is in A, then $|v| = k$ and we write $v(i) = v_i$ and $v[i] = v(1) \dots v(i)$ for $i \leq k$; so $v[0] = \epsilon$. The prefix relation is denoted $\sqsubseteq$: the finite word u is a prefix of the finite word v (denoted $u \sqsubseteq v$) if and only if there exists a (finite) word w such that $v = uw$. the finite word u is a prefix of the ω-word x (denoted $u \sqsubseteq x$) iff there exists an ω-word y such that $x = uy$.

Given a finite alphabet A, we write $A^{\leq\omega}$ for $A^* \cup A^\omega$.

Definition 2.1. *Let A be any finite alphabet, $\twoheadleftarrow \notin A$, $B = A \cup \{\twoheadleftarrow\}$, and $x \in B^{\leq\omega}$, then*
$x^{\twoheadleftarrow}$ is inductively defined by:
$\epsilon^{\twoheadleftarrow} = \epsilon$, and for a finite word $u \in (A \cup \{\twoheadleftarrow\})^*$:
$(ua)^{\twoheadleftarrow} = u^{\twoheadleftarrow} a$, if $a \in A$,
$(u\twoheadleftarrow)^{\twoheadleftarrow} = u^{\twoheadleftarrow}$ with its last letter removed if $|u^{\twoheadleftarrow}| > 0$,
$(u\twoheadleftarrow)^{\twoheadleftarrow}$ is undefined if $|u^{\twoheadleftarrow}| = 0$,
and for u infinite:

$(u)^{\twoheadleftarrow} = \lim_{n\in\omega}(u[n])^{\twoheadleftarrow}$, *where, given* β_n *and* v *in* A^*,
$v \sqsubseteq \lim_{n\in\omega} \beta_n \leftrightarrow \exists n \forall p \geq n \quad \beta_p[|v|] = v$.

We now make easy this definition to understand by describing it informally. For $x \in B^{\leq\omega}$, $x^{\twoheadleftarrow}$ denotes the string x, once every $\twoheadleftarrow$ occurring in x has been "evaluated" to the back space operation (the one familiar to your computer!), proceeding from left to right inside x. In other words $x^{\twoheadleftarrow} = x$ from which every interval of the form "$a\twoheadleftarrow$" ($a \in A$) is removed. By convention, we assume $(u\twoheadleftarrow)^{\twoheadleftarrow}$ is undefined when $u^{\twoheadleftarrow}$ is the empty sequence. i.e. when the last letter $\twoheadleftarrow$ cannot be used as an eraser (because every letter of A in u has already been erased by some eraser $\twoheadleftarrow$ placed in u). We remark that the resulting word $x^{\twoheadleftarrow}$ may be finite or infinite.

For instance,

- if $u = (a\twoheadleftarrow)^n$, for $n \geq 1$, or $u = (a\twoheadleftarrow)^\omega$ then $(u)^{\twoheadleftarrow} = \epsilon$,
- if $u = (ab\twoheadleftarrow)^\omega$ then $(u)^{\twoheadleftarrow} = a^\omega$,
- if $u = bb(\twoheadleftarrow a)^\omega$ then $(u)^{\twoheadleftarrow} = b$,
- if $u = \twoheadleftarrow(a\twoheadleftarrow)^\omega$ or $u = a\twoheadleftarrow\twoheadleftarrow a^\omega$ or $u = (a\twoheadleftarrow\twoheadleftarrow)^\omega$ then $(u)^{\twoheadleftarrow}$ is undefined.

Definition 2.2. *For* $X \subseteq A^\omega$,

$$X^{\approx} = \{x \in (A \cup \{\twoheadleftarrow\})^\omega : \ x^{\twoheadleftarrow} \in X\}.$$

The following result easily follows from [Dup01] and was applied in [Fin01,Fin04] to study the ω-powers of finitary context free languages.

Theorem 2.3. *Let* n *be an integer* ≥ 2 *and* $X \subseteq A^\omega$ *be a* $\mathbf{\Pi^0_n}$*-complete set. Then* $X^{\approx}$ *is a* $\mathbf{\Pi^0_{n+1}}$*-complete subset of* $(A \cup \{\twoheadleftarrow\})^\omega$.

Next remarks will be essential later.

Remark 2.4. Consider the following function:

$$f : \ x \in (A \cup \{\twoheadleftarrow\})^\omega \mapsto y \in A^\omega$$

defined by:

- $y = 0^\omega$ if $x^{\twoheadleftarrow}$ is finite or undefined,
- $y = x^{\twoheadleftarrow}$ otherwise.

It is clearly Borel. In fact a quick computation shows that the inverse image of any basic clopen set is Borel of low finite rank.

Remark 2.5. Let X be any subset of the Cantor space $\{0,1\}^\omega$, and f as in remark 2.4. If $0^\omega \notin X$, then for any $x \in \{0,1,\twoheadleftarrow\}^\omega$

$$x \in X^{\approx} \iff f(x) \in X$$

In other words, $X^{\approx} = f^{-1}X$. In particular, if X is Borel, so is $X^{\approx}$

3 Increasing sequences of erasers

The following construction has been partly used by the second author in [Fin04] to construct a Borel set of infinite rank which is an ω-power, i.e. in the form V^ω, where V is a set of finite words over a finite alphabet Σ. We iterate the operation $X \mapsto X^{\approx}$ finitely many times, and take the limit. More precisely,

Definition 3.1. *Given any set $X \subseteq A^\omega$:*

- $X_k^{\approx 0} = X$,
- $X_k^{\approx 1} = X^{\approx}$,
- $X_k^{\approx 2} = (X_k^{\approx 1})^{\approx}$,
- $X_k^{\approx (k)} = (X_k^{\approx (k-1)})^{\approx}$, *where we apply k times the operation $X \mapsto X^{\approx}$ with different new letters $\twoheadleftarrow_k$, $\twoheadleftarrow_{k-1}$, …, $\twoheadleftarrow_2$, $\twoheadleftarrow_1$, in such a way that we have successively:*
  - $X_k^{\approx 0} = X \subseteq A^\omega$,
  - $X_k^{\approx 1} \subseteq (A \cup \{\twoheadleftarrow_k\})^\omega$,
  - $X_k^{\approx 2} \subseteq (A \cup \{\twoheadleftarrow_k, \twoheadleftarrow_{k-1}\})^\omega$,
  - $X_k^{\approx (k)} \subseteq (A \cup \{\twoheadleftarrow_1, \twoheadleftarrow_2, \ldots, \twoheadleftarrow_k\})^\omega$.
- *We set $X^{\approx (k)} = X_k^{\approx (k)}$.*

$X^{\approx \infty} \subseteq (A \cup \{\twoheadleftarrow_n : \ 0 < n < \omega\})^\omega$ *is defined by*

$$x \in X^{\approx \infty} \iff_{def}$$

- *for each integer n, $x_n = x^{\twoheadleftarrow_1 \cdots \twoheadleftarrow_{n-1} \twoheadleftarrow_n}$ is defined, infinite, and*
- $x_\infty = \lim_{n<\omega} x_n$ *is defined, infinite, and belongs to X.*

Remark 3.2. Consider the following sequence of functions:

- $f_0(x) = x$ (f_0 is the identity),
- $f_{k+1} : (A \cup \{\twoheadleftarrow_n : k < n < \omega\})^\omega \longmapsto (A \cup \{\twoheadleftarrow_n : k+1 < n < \omega\})^\omega$ defined by:
  - $f_{k+1}(x) = x^{\twoheadleftarrow_{k+1}}$ if $x^{\twoheadleftarrow_{k+1}}$ is infinite,
  - $f_{k+1}(x) = 0^\omega$ if $x^{\twoheadleftarrow_{k+1}}$ is finite or undefined,

By induction on k, one shows that every function f_k is Borel - and even Borel of finite rank.

Moreover, since Borel functions are closed under taking the limits [Kur61], the following function is Borel.

$$f_\infty : (A \cup \{\twoheadleftarrow_n : 0 < n < \omega\})^\omega \longmapsto A^\omega$$

defined by:

- $f_\infty(x) = \lim_{n<\omega} f_n(x)$ if $\lim_{n<\omega} f_n(x)$ is defined, and infinite,
- $f_\infty(x) = 0^\omega$ otherwise.

Remark 3.3. Let $X \subset \{0,1\}^\omega$ with $0^\omega \notin X$, then for any

$$x \in (\{0,1\} \cup \{\twoheadleftarrow_n : 0 < n < \omega\})^\omega$$

$$x \in X^{\approx\infty} \iff f_\infty(x) \in X$$

In other words, $X^{\approx\infty} = {f_\infty}^{-1}(X)$, which shows that whenever X is Borel, $X^{\approx\infty}$ is Borel too.

In fact, with tools described in [Dup01], and [Dup0?], it is possible to show that given any Π^0_1-complete set Y, the set $Y^{\approx\infty}$ belongs to $\Pi^0_{\omega+2}$. If X is the set of infinite words over the alphabet $\{0,1\}$ which contains an infinite number of 1s, then it is also possible to show that $X^{\approx\infty}$ is Borel by completely different methods involving decompositions of ω-powers [FS03,Fin04].

Proposition 3.4. *Let X be the set of infinite words over $\{0,1\}$ that contain infinitely many 1s,*

$$X^{\approx\infty} \in \Delta^1_1 \setminus \Delta^0_\omega$$

Proof. The fact $X^{\approx\infty}$ is Borel is Remark 3.3. As for $X^{\approx\infty} \notin \Delta^0_\omega$, it is a consequence of the fact that the operation $Y \longmapsto Y^{\approx}$ is strictly increasing (for the Wadge ordering) inside Δ^0_ω (see [Dup01][Dup0?]). In other words, for any $Y \in \Delta^0_\omega$ the relation $Y <_W Y^{\approx}$ holds ($<_W$ stands for the strict Wadge ordering). But, as a matter of fact, $(X^{\approx\infty})^{\approx} \leq_W X^{\approx\infty}$ holds which forbids $X^{\approx\infty}$ to belong to Δ^0_ω.

Indeed, to see that $(X^{\approx\infty})^{\approx} \leq_W X^{\approx\infty}$ holds, it is enough to describe a winning strategy for player II in the Wadge game $W\left((X^{\approx\infty})^{\approx}, X^{\approx\infty}\right)$. In this game, player II uses ω many different erasers: $\twoheadleftarrow_1, \twoheadleftarrow_2, \twoheadleftarrow_3, \ldots$ whose strength is opposite to their indices ($\twoheadleftarrow_k$ erases all erasers $\twoheadleftarrow_j$ for any $j > k$ but no $\twoheadleftarrow_i$ for $i \leq k$). While player I uses the same erasers as player II does, plus an extra one ($\twoheadleftarrow$) which is stronger than all the other ones.

The winning strategy for II derives from ordinal arithmetic: $1 + \omega = \omega$. It consists in copying I's run with a shift on the indices of erasers:

- if I plays a letter 0 or 1, then II plays the same letter,
- if I plays an eraser $\twoheadleftarrow_n$, II plays the eraser $\twoheadleftarrow_{n+1}$.
- if I plays the eraser $\twoheadleftarrow$ (the first one that will be taken into account when the erasing process starts), then II plays $\twoheadleftarrow_0$.

This strategy is clearly winning.

4 Simulating $X^{\approx\infty}$ by the ω-power of a context-free language

It was already known that there exists an ω-power of a finitary language which is Borel of infinite rank [Fin04]. But the question was left open whether such a finitary language could be *context free*.

This article provides effectively a context free language V such that V^ω is a Borel set of infinite rank, and uses infinite Wadge games to show that this ω-power V^ω is located above $\mathbf{\Delta}^0_\omega$ in the Borel hierarchy.

The idea is to have $X^{\approx\infty}$, where X stands for the set of all infinite words over $\{0,1\}$ that contain infinitely many 1s to be of the form V^ω for some language V recognized by a (non deterministic) Pushdown Automaton. We first recall the notion of pushdown automaton [Ber79,ABB96].

Definition 4.1. *A pushdown automaton (PDA) is a 7-tuple*

$$M = (Q, A, \Gamma, \delta, q_0, Z_0, F)$$

where

- Q *is a finite set of states,*
- A *is a finite input alphabet,*
- Γ *is a finite pushdown alphabet,*
- $q_0 \in Q$ *is the initial state,* $Z_0 \in \Gamma$ *is the start symbol,*
- δ *is a mapping from* $Q \times (A \cup \{\varepsilon\}) \times \Gamma$ *to finite subsets of* $Q \times \Gamma^*$.
- $F \subseteq Q$ *is the set of final states.*

If $\gamma \in \Gamma^+$ *describes the pushdown store content, the leftmost symbol of* γ *will be assumed to be on "top" of the store. A configuration of a PDA is a pair* (q, γ) *where* $q \in Q$ *and* $\gamma \in \Gamma^*$.

For $a \in A \cup \{\varepsilon\}$, $\gamma, \beta \in \Gamma^*$ *and* $Z \in \Gamma$, *if* (p, β) *is in* $\delta(q, a, Z)$, *then we write* $a : (q, Z\gamma) \mapsto_M (p, \beta\gamma)$.

$\mapsto_M^*$ *is the transitive and reflexive closure of* $\mapsto_M$.

Let $u = a_1 a_2 \dots a_n$ *be a finite word over* A. *A finite sequence of configurations* $r = (q_i, \gamma_i)_{1 \leq i \leq p}$ *is called a run of* M *on* u, *starting in configuration* (p, γ), *iff:*

1. $(q_1, \gamma_1) = (p, \gamma)$
2. *for each* i, $1 \leq i \leq p-1$, *there exists* $b_i \in A \cup \{\varepsilon\}$ *satisfying* $b_i : (q_i, \gamma_i) \mapsto_M (q_{i+1}, \gamma_{i+1})$ *such that* $a_1 a_2 \dots a_n = b_1 b_2 \dots b_{p-1}$.

This run is simply called a run of M *on* u *if it starts from configuration* (q_0, Z_0).

The language accepted by M *is*

$$L(M) = \{u \in A^* : \text{there is a run } r \text{ of } M \text{ on } u \text{ ending in a final state}\}.$$

For instance, the set $0^*1 \subset \{0,1\}^*$ is trivially context-free.

Proposition 4.2 (Finkel). *Let* L_n *be the maximal subset of* $\{0, 1, \twoheadleftarrow_1, \twoheadleftarrow_2, \dots, \twoheadleftarrow_n\}^*$ *such that* $L_n^{\twoheadleftarrow_1 \twoheadleftarrow_2 \cdots \twoheadleftarrow_n} = 0^*1$,

$$L_n \text{ is context-free}$$

This was first noticed by the second author in [Fin01].

To be more precise, by $u \in L_n$ we mean: we start with some u, then we evaluate $\twoheadleftarrow_1$ as an eraser, and obtain u_1 (providing that we must never use $\twoheadleftarrow_1$ to erase the empty sequence, i.e. every occurrence of a $\twoheadleftarrow_1$ symbol does erase a letter 0 or 1 or an eraser $\twoheadleftarrow_i$ for $i > 1$). Then we

start again with u_1, this time we evaluate $\twoheadleftarrow_2$ as an eraser, which yields u_2, and so on. When there is no more symbol $\twoheadleftarrow_i$ to be evaluated, we are left with $u_n \in \{0,1\}^*$. We define $u \in L_n$ iff $u_n \in 0^*1$.

To make a PDA recognize L_n, the idea is to have it guess (non deterministically), for each single letter that it reads, whether this letter will be erased later or not. Moreover, the PDA should also guess for each eraser it encounters, whether this eraser should be used as an eraser or whether it should not - for the only reason that it will be erased later on by a *stronger* eraser. During the reading, the stack should be used to accumulate all pendant guesses, in order to verify later on that they are fulfilled.

We would very much like to prove that $L_\infty = \bigcup_{n<\omega} L_n$ is context-free. Unfortunately, we cannot get such a result. However, we are able to show that a slightly more complicated set (strictly containing L_∞) is indeed context-free.

Of course, the first problem that comes to mind when working with L_∞, is to handle ω many different erasers with a finite alphabet. This implies that erasers must be coded by finite words. This was done by the second author in [Fin03b]. Roughly speaking, the eraser $\twoheadleftarrow_n$ is coded by the word $\alpha B^n C^n D^n E^n \beta$ with new letters $\alpha, B, C, D, E, \beta$. It is a little bit tricky, but the PDA must really be able to read the number n identifying the eraser four times.

The very definition of the sets L_n, requires the erasing operations to be executed in an increasing order: in a word that contains only the erasers $\twoheadleftarrow_1, \ldots, \twoheadleftarrow_n$, one must consider first the eraser $\twoheadleftarrow_1$, then $\twoheadleftarrow_2$, and so on...

Therefore this erasing process satisfy the following properties:

(a) An eraser $\twoheadleftarrow_j$ may only erase letters $c \in \{0,1\}$ or erasers $\twoheadleftarrow_k$ with $k > j$.

(b) Assume that in a word $u \in L_n$, there is a sequence cvw where c is either in $\{0,1\}$ or in the set $\{\twoheadleftarrow_1, \ldots, \twoheadleftarrow_{n-1}\}$, and w is (the code of) an eraser $\twoheadleftarrow_k$ which erases c once the erasing process is achieved. If there is in v (the code of) an eraser $\twoheadleftarrow_j$ which erases e, where $e \in \{0,1\}$ or e is (the code of) another eraser, then e must belong to v (it is between c and w in the word u) ; moreover the erasing - by the eraser $\twoheadleftarrow_j$ - has been achieved before the other one with the eraser

$\twoheadleftarrow_k$. This implies $j \leq k$. Thus the integer k must satisfy:

$$k \geq \max\{j : \text{ an eraser } \twoheadleftarrow_j \text{ was used inside } v\}$$

The essential difference with the case studied in [Fin03b] is that here an eraser $\twoheadleftarrow_j$ may only erase letters 0 or 1 or erasers $\twoheadleftarrow_k$ for $k > j$, while in [Fin03b] an eraser $\twoheadleftarrow_j$ was assumed to be only able to erase letters 0 or 1 or erasers $\twoheadleftarrow_k$ for $k < j$. So the above inequality was replaced by:

$$k \leq \min\{j : \text{ an eraser } \twoheadleftarrow_j \text{ was used inside } v\}$$

However, with a slight modification, we can construct a PDA $\mathcal{B}$ which, among words where letters $\alpha, \beta, B, C, D, E$ are only used to code erasers of the form $\twoheadleftarrow_j$, accepts exactly the words which belong to the language L_∞. We now explain the behavior of this PDA. (For simplicity, we sometimes talk about the eraser $\twoheadleftarrow_j$ instead of its code $\alpha B^j C^j D^j E^j \beta$.)

Assume that $\mathcal{A}$ is a finite automaton accepting (by final state) the finitary language 0^*1 over the alphabet $A = \{0, 1\}$.
We can informally describe the behavior of the PDA $\mathcal{B}$ when reading a word u such that the letters $\alpha, B, C, D, E, \beta$ are only used in u to code the erasers $\twoheadleftarrow_j$ for $1 \leq j$.

$\mathcal{B}$ simulates the automaton $\mathcal{A}$ until it guesses (non deterministically) that it begins to read a segment w which contains erasers which really erase and some letters of A or some other erasers which are erased when the operations of erasing are achieved in u.

Then, still non deterministically, when $\mathcal{B}$ reads a letter $c \in A$ it may guess that this letter will be erased and push it in the pushdown store, keeping in memory the current state of the automaton $\mathcal{A}$.

In a similar manner when $\mathcal{B}$ reads the code $\twoheadleftarrow_j = \alpha B^j C^j D^j E^j \beta$, it may guess that this eraser will be erased (by another eraser $\twoheadleftarrow_k$ with $k < j$) and then may push in the store the finite word $\gamma E^j \nu$, where γ, E, ν are in the pushdown alphabet of $\mathcal{B}$.

But $\mathcal{B}$ may also guess that the eraser $\twoheadleftarrow_j = \alpha B^j C^j D^j E^j \beta$ will really be used as an eraser. If it guesses that the code of $\twoheadleftarrow_j$ will be used as an eraser, $\mathcal{B}$ has to pop from the top of the pushdown store either a letter $c \in A$ or the code $\gamma E^i.\nu$ of another eraser $\twoheadleftarrow_i$, with $i > j$, which is erased by $\twoheadleftarrow_j$.

In this case, it is easy for $\mathcal{B}$ to check whether $i > j$ when reading the initial segment αB^j of $\twoheadleftarrow_j$.

But as we remarked in (b), the PDA $\mathcal{B}$ must also check that the integer j is greater than or equal to every integer p such that an eraser $\twoheadleftarrow_p$ has been used since the letter $c \in A$ or the code $\gamma E^i.\nu$ was pushed in the store. Then, after having pushed some letter $t \in A$ or the code $t = \gamma E^i.\nu$ of an eraser in the pushdown store, and before popping it from the top of the stack, $\mathcal{B}$ must keep track of the following integer in the memory stack.

$k = max[p \,/\,$ some eraser $\twoheadleftarrow_p$ has been used since t was pushed in the stack $]$

For that purpose $\mathcal{B}$ pushes the finite word $L_2 S^k L_1$ in the pushdown store (L_1 is pushed first, then S^k and the letter L_2), with L_1, L_2 and S are new letters added to the pushdown alphabet.

So, when $\mathcal{B}$ guesses that $\twoheadleftarrow_j = \alpha B^j C^j D^j E^j \beta$ will be really used as an eraser, there is on top of the stack either a letter $c \in A$ or a code $\gamma E^i.\nu$ of an eraser which will be erased or a code $L_2 S^k L_1$. The behavior of $\mathcal{B}$ is then as follows.

Assume first there is a code $L_2 S^k L_1$ on top of the stack. Then $\mathcal{B}$ firstly checks that $j \geq k$ holds by reading the segment $\alpha B^j C$ of the eraser $\alpha B^j C^j D^j E^j \beta$.

If $j \geq k$ holds, then using ϵ-transitions, $\mathcal{B}$ completely pops the word $L_2 S^k L_1$ from the top of the stack. ($\mathcal{B}$ has already checked it is allowed to use the eraser $\twoheadleftarrow_j$).

Then, in each case, the top of the stack contains either a letter $c \in A$, or the code $\gamma E^i \nu$ of an eraser which should be erased later. $\mathcal{B}$ pops this letter c or the code $\gamma E^i.\nu$ (having checked that $j < i$ after reading the segment $\alpha B^j C^j$ of the eraser $\alpha B^j C^j D^j E^j \beta$).

A this point, we must have a look at the top stack symbols. There are three cases:

1. The top stack symbol is the bottom symbol Z_0. In which case, the PDA $\mathcal{B}$, after having completely read the eraser $\twoheadleftarrow_j$, may pursue the simulation of the automaton $\mathcal{A}$ or guess that it begins to read another segment v which will be erased. Hence the next letter $c \in A$ or the next code $\alpha B^m.C^m.D^m.E^m.\beta$ of the word will be erased. Then $\mathcal{B}$ pushes the letter $c \in A$ or the code $\gamma E^m.\nu$ of $\twoheadleftarrow_m$ in the pushdown store.

2. If the top stack symbol is either a letter $c' \in A$ or a code $\gamma E^m.\nu$, then $\mathcal{B}$ pushes the code $L_2 S^j L_1$ in the pushdown store (j is then the maximum of the set of integers p such that an eraser $\twoheadleftarrow_p$ has been used since the letter c' or the code $\gamma E^m.\nu$ has been pushed into the stack).
3. If the top stack symbols are a code $L_2 S^l L_1$, then the PDA $\mathcal{B}$ must compare the integers j and l, and replace $L_2 S^l L_1$ by $L_2 S^j L_1$ in case $j > l$. $\mathcal{B}$ achieves this task while reading the segment $D^j E^j \beta$ of the eraser $\alpha B^j C^j D^j E^j \beta$.
   The PDA $\mathcal{B}$ pops a letter S for each letter D it reads. Then it checks whether $j \geq l$ is satisfied.
   If $j \geq l$ then it pushes $L_2 S^j L_1$ while it reads the segment $E^j \beta$ of the eraser $\twoheadleftarrow_j$.
   In case $j < l$, after it reads D^j, the part $S^{l-j} L_1$ of the code $L_2 S^l L_1$ remains in the stack. The PDA then pushes again j letters S and a letter L_2 while reading $E^j \beta$.

When again the stack only contains Z_0 - the initial stack symbol - $\mathcal{B}$ resumes the simulation of the automaton $\mathcal{A}$ or it guesses that it begins to read a new segment which will be erased later.

We are confronted with the fact $\mathcal{B}$ will also accept some words where the letters $\alpha, \beta, B, C, D, E$ are not used to code erasers. How can we make sure that this PDA is not misled by such wrong codes of erasers ?

5 Wrong codes of erasers and the right ω-power

In fact, one cannot make sure that a PDA notices the discrepancy between right codes of the form $\alpha B^j C^j D^j E^j \beta$ and wrong ones (of the form $\alpha B^b C^c D^d E^e \beta$ where b, c, d, e are not all the same integer for instance). However, there is a satisfactory solution: instead of having a PDA reject these wrong codes, simply let it accept all of them. Accepting a word if it contains a wrong code of an eraser is trivial for a non deterministic PDA. So instead of a PDA $\mathcal{B}$ that accepts precisely L_∞ (up to the coding of erasers), we set

Proposition 5.1. *There exists a PDA $\mathcal{B}$ s.t.*

$$\mathcal{L}(\mathcal{B}) = L_\infty \cup W$$

where W stands for the set of all finite words which host a wrong code, L_∞ really is L_∞ where erasers are replaced by their correct codes, and $\mathcal{L}(\mathcal{B})$ is the language recognized by $\mathcal{B}$. Everything is ready for the main result.

Theorem 5.2. *The ω-power $Y = \mathcal{L}(\mathcal{B})^\omega$ of the context-free language $L(\mathcal{B})$ described above satisfies*

$$Y \in \Delta^1_1 \setminus \Delta^0_\omega$$

Proof. To begin with, the set Y is the disjoint union of three different sets: $Y = Y_0 \cup Y_\infty \cup Y_*$, where Y_0 is the set of all infinite sequences in Y with no wrong code in them, Y_∞ the set of all infinite sequences with infinitely many wrong codes, and Y_* the set of infinite sequences with finitely many wrong codes (at least one). We remark that:

- Y_0 is Wadge equivalent to the set $X^{\approx\infty}$ as defined in 3.4. i.e. the set of all ω-words that, after taking care of the erasing process, ultimately reduce to words with infinitely many 1s. To be more precise, it is this very same set up to a renaming of the erasers. So Y_0 belongs to Δ^1_1.
- Y_∞ is Wadge equivalent to X, so it is Π^0_2-complete.
- Y_* is more complicated. However, it is of the form $Y_* = WY_0$, where W is the set of all finite words with at least an occurrence of a wrong code. So Y_* is a countable union of sets, each of which is Wadge equivalent to Y_0. Hence, Y_* is a countable union of Borel sets, therefore Y_* is Borel too.

All three cases put together show that $Y = Y_0 \cup Y_\infty \cup Y_*$, is a finite union of Borel sets, hence it Borel too.

It remains to prove that $Y \notin \Delta^0_\omega$. This, in fact, is immediate from Proposition 3.4 which stated that $X^{\approx\infty} \notin \Delta^0_\omega$. Because there is an obvious winning strategy for player II in the Wadge game $W(X^{\approx\infty}, Y)$. It consists in never playing a wrong code, and copying I's run up to the renaming of the erasers. Since $X^{\approx\infty}$ is clearly Wadge equivalent to Y_0 this strategy works perfectly well and shows that $Y \notin \Delta^0_\omega$.

This quick study gives an example of how an infinite game theoretical approach leads to intriguing results in Theoretical Computer Science. On one hand, the notion of erasers is highly related to the dynamic behavior of players in games. And, on the other hand, non determinism provides

very effective ways to deal with the erasing process. So, all together, they afford a method for describing (topological) complexity of very effective sets of reals.

Acknowledgments. we wish to thank an anonymous referee for useful comments on a previous version of this paper.

References

ABB96. Jean-Michel **Autebert**, Jean **Berstel** and Luc **Boasson**, Context Free Languages and Pushdown Automata, in Handbook of Formal Languages, Vol 1, Springer Verlag 1996.

Ber79. Jean **Berstel**, Transductions and Context Free Languages, Teubner Studienb"ucher Informatik, 1979.

Dup01. Jacques **Duparc**, Wadge hierarchy and Veblen hierarchy. Part 1 : Borel sets of finite rank, Journal of Symbolic Logic Vol. 66, 1 (2001), p. 56–86

Dup0?. Jacques **Duparc**, Wadge hierarchy and Veblen hierarchy. Part 2 : Borel sets of infinite rank, submitted to the Journal of Symbolic Logic. Available online on " http://www-iis.unil.ch∼jduparc/"

Fin01. Olivier **Finkel**, Topological Properties of Omega Context Free Languages, Theoretical Computer Science, Vol. 262 (1-2), July 2001, p. 669-697.

Fin03a. Olivier **Finkel**, Borel Hierarchy and Omega Context Free Languages, Theoretical Computer Science, Vol. 290 (3), 2003, p. 1385-1405.

Fin03b. Olivier **Finkel**, On Omega Context Free Languages which are Borel Sets of Infinite Rank, Theoretical Computer Science, Vol. 299 (1-3), 2003, p. 327-346.

Fin04. Olivier **Finkel**, An ω-Power of a Finitary Language which is a Borel Set of Infinite Rank, Fundamenta Informaticae, Vol. 62 (3-4), 2004, p. 333-342.

FS03. Olivier **Finkel** and Pierre **Simonnet**, Topology and Ambiguity in Omega Context Free Languages, Bulletin of the Belgian Mathematical Society, Vol. 10 (5), 2003, p. 707-722.

HU69. John E. **Hopcroft** and Jeffrey D. **Ullman**, Formal Languages and their Relation to Automata, Addison-Wesley Publishing Company, Reading, Massachussetts, 1969.

Kec94. Alexander **Kechris** *Classical descriptive set theory.* Graduate texts in mathematics; vol 156. Springer Verlag (1994)

Kur61. Casimir **Kuratowski** Topologie I et II, tome I, 4e dition, 1958 et tome II, 3e dition, 1961, Reprint, 1992 Editions Jacques Gabay, 24,5 x 18 oblong, 528 p., BrochZ, 2 tomes en 1 volume, ISBN 2-87647-141-8.

Lec01. Dominique **Lecomte**, Sur les Ensembles de Phrases Infinies Constructibles a Partir d'un Dictionnaire sur un Alphabet Fini, Séminaire d'Initiation a l'Analyse, Volume 1, année 2001-2002.

Lec05. Dominique **Lecomte**, Omega-Powers and Descriptive Set Theory, Journal of Symbolic Logic, Volume 70 (4), 2005, p. 1210-1232.

LT94. Helmut **Lescow** and Wolfgang **Thomas**, Logical Specifications of Infinite Computations, In: "A Decade of Concurrency (J. W. de Bakker et al., eds), Springer LNCS 803 (1994), 583-621.

Lou83. Alain **Louveau** *Some Results in the Wadge Hierarchy of Borel Sets.* Cabal Sem 79-81, Lecture Notes in Mathematics (1019) 28-55. (1983)

Mar75. Donald A. **Martin** *Borel Determinacy.* Ann. Math. vol.102 (1975) 363-371.

Mos80. Yiannis N. **Moschovakis**, Descriptive Set Theory, North-Holland, Amsterdam 1980.

Niw90. Damian **Niwinski**, Problem on ω-Powers posed in the Proceedings of the 1990 Workshop "Logics and Recognizable Sets" (Univ. Kiel).

PerPin04. Dominique **Perrin**, Jean-Éric **Pin** (*eds.*), Infinite Words. Automata, Semigroups, Logic and Games Amsterdam 2004

Sim92. Pierre **Simonnet**, Automates et Théorie Descriptive, Ph. D. Thesis, Université Paris 7, March 1992.

Sta86. Ludwig **Staiger**, Hierarchies of Recursive ω-Languages, Jour. Inform. Process. Cybernetics EIK 22 (1986) 5/6, 219-241.

Sta97a. Ludwig **Staiger**, ω-Languages, Chapter of the Handbook of Formal Languages, Vol 3, edited by G. Rozenberg and A. Salomaa, Springer-Verlag, Berlin, 1997.

Sta97b. Ludwig **Staiger**, On ω-Power Languages, in New Trends in Formal Languages, Control, Coperation, and Combinatorics, Lecture Notes in Computer Science 1218, Springer-Verlag, Berlin 1997, 377-393.

Tho90. Wolfgang **Thomas**, Automata on Infinite Objects, in: J. Van Leeuwen, ed., Handbook of Theoretical Computer Science, Vol. B (Elsevier, Amsterdam, 1990), p. 133-191.

Veb08. Oswald **Veblen** *Continuous increasing fonctions of finite and transfinite ordinals.* Transactions of the American Mathematical Society vol 9 (1908) 280-292.

Wad72. William W. **Wadge** *Degrees of complexity of subsets of the Baire space.* Notice A.M.S. (1972), A-714

Wad84. William W. **Wadge** *Reducibility and determinateness on the Baire space.* Ph.D. Thesis, Berkeley.

vW78. Robert A. **van Wesep** *Wadge degrees and descriptive set theory* Cabal Seminar 76–77 (Proc. Caltech-UCLA Logic Sem., 1976–77), pp. 151–170, Lecture Notes in Math., 689, Springer, Berlin, (1978).